Analyse et Conception

12. Les Patrons GoF

Patrons «GoF» Les Pattrons selon GoF

"Gang of Four" (aka GoF) :

E.Gamma, R. Helm, R. Johnson & J. Vlissides,(1995) *Design Patterns: Elements of Reusable Software*, Addison–Wesley.

« GoF » ont introduit l'idée de « *design pattern* » en génie logiciel OO:

> **«** A pattern can only be a pattern if it can be shown to exist in many different and distinct items of software**.»**

Mais à l'origine, l'idée appartient à un architecte, **Christopher Alexander**

> **«** A pattern is a **three-part rule**, which expresses a relation between **a certain context, a problem,** and **a solution. »**

Patrons «GoF» — Une Règle Tri-Partie

- La relation entre un ***problème*****,** dans un certain ***contexte*** et une ***solution:***
  - ***Le problème:*** *décrit ce qui est demandé par le logiciel, c'est-à-dire la* ***difficulté*** *qu'on doit résoudre ou la* ***qualité*** *qu'on doit assurer.*
  - ***Le contexte:*** *décrit l'****environnement*** *dans lequel la solution est* ***valide.***
  - ***La solution:*** *décrit* ***les classes, les données*** *et* ***les méthodes*** *utilisées pour résoudre le problème.*

Patrons «GoF» — GoF: Mode d'emploi

- **On reconnait le problème à un niveau élevé**
  - *i.e., on le voit* ***du point de vue de l'utilisateur*** *et non du point de vue de sa* ***localisation*** *dans notre* ***application***
- **Ensuite, on conçoit la solution**
  - *on définit les* ***classes****,* ***attributs*** *et* ***méthodes*** *qui représentent le* ***comportement*** *du patron*
- **Enfin, on implémente la solution et on la teste**
  - *en utilisant les* ***tests unitaires*** *de méthodes et de classes, on s'assure que le composant qu'on a créé est* ***fonctionnel*** *et* ***réutilisable***

Patrons «GoF» — Patrons GoF

Aspect important des patrons : pas créés mais identifiés

- *Les patrons **existent** en **génie logiciel**, comme ils existent dans la **nature**,*
- *Les **livres** sur les patrons de conception sont typiquement des **catalogues** des patrons existant,*
- *Les patrons de conception sont **identifiés** en cherchant des **problèmes recurrents** et leurs **solutions**,*
- *Souvent, la longue expérience est la seule voie vers la découverte de patrons.*

Patrons «GoF» — Typologie des PC GoF

- Au moins trois types de patrons GoF :
  - **Créationnel** : liés à la création des objets,
  - **Structurel** : liés à la composition des objets,
  - **Comportemental** : liés à l'intéraction entre objets et/ou classes.
- De plus, de patrons GoF peuvent avoir une des deux portées :
  - **Classe** : les patrons de classe mettent l'accent sur les relations entre classes,
  - **Objet** : les patrons d'objet mettent l'accent sur les relations et les intéractions entre objets.

Patrons «GoF» — La famille GoF

Scope		Purpose: Creational	Purpose: Structural	Purpose: Behavioral
Class		Factory Method	Adapter (class)	Interpreter Template Method
Object	☺	Abstract Factory ☺ Builder Prototype ☺ Singleton ☺	Adapter (object) Bridge ☺ Composite Decorator Flyweight Facade ☺ Proxy ☺ ☺	Chain of Responsibility Command Iterator Mediator Memento Observer ☺ State Strategy Visitor

Objectif: reflète ce que le patron fait,

Portée: le domaine d'application (ensemble d'éléments).

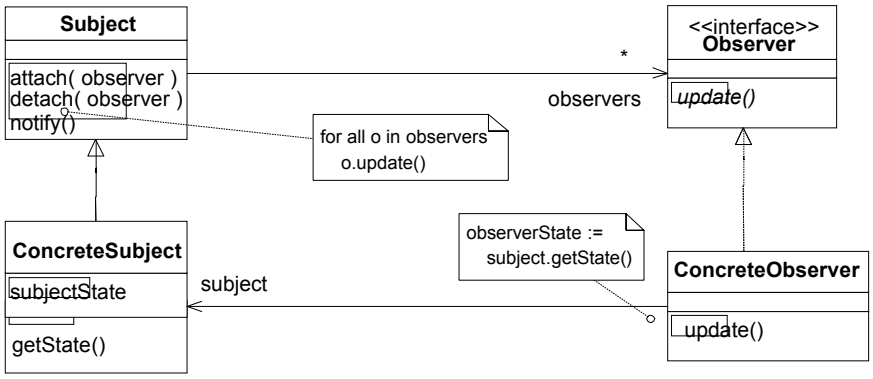
Patrons «GoF» — «Observer»

- **Problème =** Avertissement effectif des objets intéressés (**Observateurs**) par les changements survenus dans l'état d'un objet particulier (**Observé** ou **Sujet**) ?

- **Intention =** Définit une relation 1-N entre l'Observé et les Observateurs de sorte à ce que les changements d'état soient signalés automatiquement aux Observateurs (sans que l'Observé ait besoin de les connaître individuellement).

- **Applicabilité =** une abstraction a deux aspects, l'un **dépendant** de l'autre; le **changement dans un** objet **entraîne de changements dans d'autres**, sans que l'on sache combien d'autres objets doivent être changés; un objet devrait en avertir d'autres **sans** pour autant **faire des hypothèses sur la nature** de ces autres objets.

Patrons «GoF» Diagramme de Classes



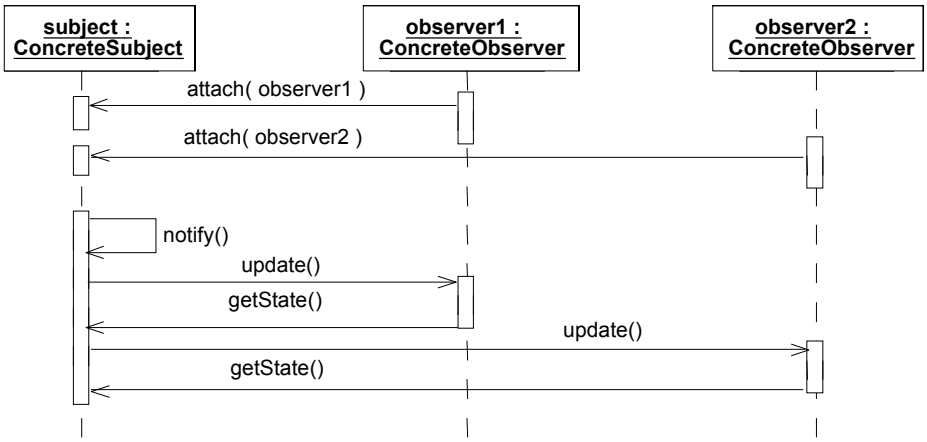
Patrons «GoF» Participants

- **Subject**
  - *Connaît les observateurs à travers une **interface** (ignore leur identité exacte, c'est-à-dire leur classe),*
  - *Fournit une **interface** pour l'ajout et l'enlèvement d'un observateur.*
- **Observer**
  - *Fournit une **interface** permettant les avertissements.*
- **ConcreteSubject**
  - *Possède un état ayant d'intérêt pour les objets de type **ConcreteObserver**.*
  - *Envoie une notification aux observateurs lors d'un changement d'état.*
- **ConcreteObserver**
  - *Responsible de maintenir son propre état en concordance avec l'état du **ConcreteSubject**.*
  - *Implémente l'interface **Observer**.*
  - *Maintient une référence vers l'objet de **ConcreteSubject** (optionnel).*

Patrons «GoF» — Collaborations

- ConcreteSubject
  - *Avertit l'ensemble des observateurs du l'avènement d'un changement dans son état .*
- ConcreteObserver
  - *Peut envoyer de requêtes au* **ConcreteSubject** *pour connaître les détails de son nouvel état.*

Patrons «GoF» — Diagr. de Séquence



Implantation

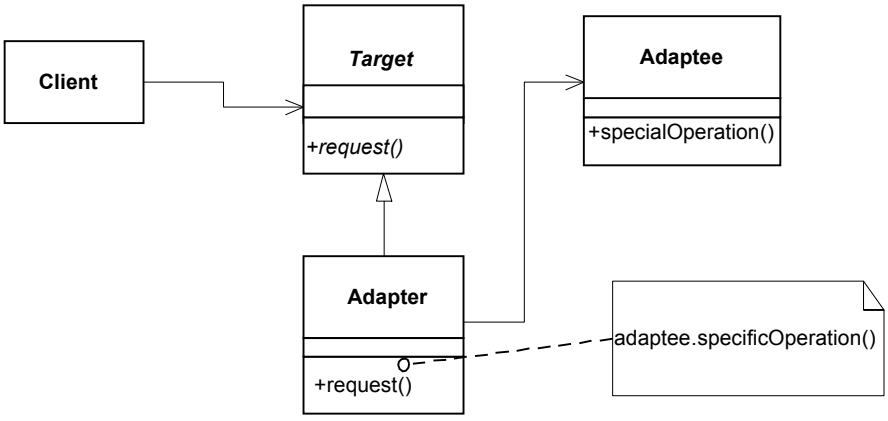
- Identifier les participants,
  - *Ex. de l'application POST*
  - *SaleFrame = ConcreteSubject*
  - *SystemController = ConcreteObserver*
- Correspondance Subject-Observer, quel modèle ?
- Références pendantes *à éviter*,
- Enregistrement explicite des modifications d'intérêt.

- ☺ **Modularité**: le Subject et les Observers peuvent évoluer de façon indépendante
- ☺ **Extensibilité**: le nombre d'Observer n'est pas limité *a priori*
- ☺ **Adaptabilité**: les Observers différents peuvent fournir de perspectives différentes (et complémentaires) sur le Subject
- ☹ Mises à jour **inattendues**: les Observers ne se connaissent pas
- ☹ **Coût** de la mise à jour: certains Observers peuvent avoir besoin d'indices sur ce qui a changé

Patrons «GoF» «Adapter»

- **Problème** = faire collaborer des classes dont les interfaces sont incompatibles.
- **Intention** = le patron **Adapter** convertit l'interface d'une classe en une autre interface que le client s'attend à retrouver. Un objet-adaptateur est inséré entre le client et la classe « adaptée ».
- **Applicabilité** = la réutilisation d'une classe existante est souhaitable, mais l'interface de cette classe ne correspond pas au besoins; conception d'une classe vouée à la réutilisation qui est obligée de collaborer avec des classes imprévues dont les interface sont incompatibles.

Patrons «GoF» Diagramme de Classes



Patrons «GoF» — Participants

- **Target**
  - *Définit l'interface spécifique au domaine que le client va utiliser.*
- **Client**
  - *Collabore avec les objets qui fournissent l'interface* **Target**.
- **Adaptee**
  - *Définit sa propre interface (déjà existante),*
  - *Sujet à adaptation par* **Adapter**.
- **Adapter**
  - *Adapte l'interface* **Adaptee** *vers l'interface* **Target**.

Patrons «GoF» — Collaborations

- Client
  - *Communique avec* Adapter *en lui envoyant des messages qui invoquent des méthodes de son interface.*
- Adapter
  - *Renvoie les requêtes vers* Adaptee *tout en les « traduisant » au préalable par des messages compréhensibles pour celui-ci.*

Patrons «GoF» — Exemple